Controller Synthesis for Linear Time-varying Systems with Adversaries

Zhenqi Huang Yu Wang Sayan Mitra Geir Dullerud

{zhuang25, yuwang8, mitras, dullerudge}@illinois.edu
Coordinate Science Laboratory
University of Illinois at Urbana Champaign
Urbana, IL 61801

ABSTRACT

We present a controller synthesis algorithm for a discrete time reach-avoid problem in the presence of adversaries. Our model of the adversary captures typical malicious attacks envisioned on cyber-physical systems such as sensor spoofing, controller corruption, and actuator intrusion. After formulating the problem in a general setting, we present a sound and complete algorithm for the case with linear dynamics and an adversary with a budget on the total L2-norm of its actions. The algorithm relies on a result from linear control theory that enables us to decompose and precisely compute the reachable states of the system in terms of a symbolic simulation of the adversary-free dynamics and the total uncertainty induced by the adversary. With this decomposition, the synthesis problem eliminates the universal quantifier on the adversary's choices and the symbolic controller actions can be effectively solved using an SMT solver. The constraints induced by the adversary are computed by solving second-order cone programmings. The algorithm is later extended to synthesize state-dependent controller and to generate attacks for the adversary. We present preliminary experimental results that show the effectiveness of this approach on several example problems.

Keywords

Cyber-physical security, constraint-based synthesis, controller synthesis

1. INTRODUCTION

We study a discrete time synthesis problem for a plant simultaneously acted-upon by a *controller* and an *adversary*. Synthesizing controller strategies for stabilization in the face of random noise or disturbances is one of the classical problem in control theory [1, 2]. Synthesis for temporal logic specifications [3–5], for discrete, continuous, and hybrid systems have been studied in detail. The reach-avoid properties that our controllers target are special, bounded-time temporal logic requirements, and they have received special attention as well [6]. Unlike the existing models in controller synthesis literature, however, the system here is afflicted by an adversary and we would like to synthesize a controller that guarantees its safety and liveness for *all* possible choices made by the adversary.

This problem is motivated by the urgent social to secure control modules in critical infrastructures and safety-critical systems against malicious attacks [7, 8]. Common modes of attack include sensor spoofing or jamming, malicious code, and actuator intrusion. Abstracting the mechanisms used to launch the attacks, their effect on physical plant can be captured as a switched system with inputs from the controller and the adversary:

$$x_{t+1} = f_{\sigma_t}(x_t, u_t, a_t),$$

where x_t is the state of the system, u_t and a_t are the inputs from the controller and the adversary. The problem is parameterized by a family of dynamical functions $\{f_\sigma\}_{\sigma\in\Sigma}$, a switching signal $\{\sigma_t\}_{t\in\mathbb{N}}$, a time bound T, the set of initial sates (*Init*), target states (*Goal*), safe states (*Safe*), the set of choices available to the adversary (*Adv*) and the controller (*Ctr*). A natural decision problem is to ask: Does there exist a controller strategy $\mathbf{u} \in Ctr$ such that for any initial state in *Init*, and any choice by the adversary in *Adv* the system remains *Safe* and reaches *Goal* within time T. A constructive affirmative answer can be used to implement controllers that are *Adv*-resilient, while a negative answer can inform the system design choices that influence the other parameters like f, T and Ctr.

We provide a decision procedure for this problem for the special case where f is a linear mapping, the sets *Init*, *Safe*, *Goal*, and *Ctr* sets are given as by polytopic sets and *Adv* is given as an ℓ^2 ball in an Euclidean space. The idea behind the algorithm is a novel decomposition that distinguishes it from the LTL-based synthesis approaches [3] and reachability-based techniques of [6]. The key to this decomposition is the concept of *adversarial leverage*: the uncertainty in the state of the system induced by the sequence of choices made by the adversary, for a given initial state and a sequence of choices made by the controller. For linear models, we show that the adversary leverage can be computed exactly. As a result, an adversary-free synthesis problem with a modified set of *Safe* and *Goal* requirements, precisely

gives the solution for the problem with adversary.

We implement the algorithm with a convex optimization package CVXOPT [9] and an SMT solver Z3 [10]. We present experimental results that show the effectiveness of this approach on several example problems. The algorithm synthesizes adversary-resilient control for systems with up to 16 dimensions in minutes. We have that the algorithm can be applied to to analyze the maximum power of the adversary such that a feasible solution exists and to synthesize attacks for adversary.

Advancing Science of Security.

Scientific security analysis is necessarily parameterized by the the skill and effort level of the adversary. In this paper we combine these parameters into a single parameter called the *budget* of the adversary which can model sensor attacks and actuator intrusions with different strengths and persistence. We present the foundations for analyzing cyber-physical systems under attack from these adversaries with different budgets. Specifically, we develop algorithms for both automatic synthesis of safe controllers and for proving that there exists no satisfactory controller, when the adversary has a certain budget. These algorithms can be also used to characterize vulnerability of system states in terms of the adversary budget that make them infeasible for safe control. In summary, we present a framework for algorithmically studying security of cyberphysical systems in the context of model-based development.

2. RELATED WORK

In this work, we employ SMT solvers to synthesize controllers for reach-avoid problems for discrete-time linear systems with adversaries. Our problem is formulated along the line of the framework and fundamental design goals of [7,11]. The framework was applied to study optimal control design with respect a given objective function under security constraints [12] and the detection of computer attacks with the knowledge of the physical system [13]. Similar frameworks were adopted in [14] where the authors proposed an effective algorithm to estimate the system states and designed feedback controllers to stabilize the system under adversaries, and in [15] where a optimal controller is designed for a distributed control system with communication delays. Although the motivation of the above studies are similar to ours, we focus on another aspect of the problem which is to synthesize attack-resilient control automatically.

The idea of using SMT solvers to synthesize feedback controllers for control systems is inspired by recent works [16, 17]. In [16], the authors used SMT solvers to synthesize integrated task and motion plans by constructing a placement graph. In [17], a constraint-based approach was developed to solve games on infinite graphs between the system and the adversary. Our work extend the idea of constraint-based synthesis by introducing control theoretic approaches to derived the constraints.

The authors of [6, 18] proposed a game theoretical approach to synthesize controller for the reach-avoid problem, first for continuous and later for switched systems. In these approaches, the reach set of the system is computed by solving a non-linear Hamilton-Jacobi-Isaacs PDE. Our methodology, instead of formulating a general optimization problem for which the solution may not be easily computable, solves a special case exactly and efficiently. With this building block, we are able to solve more general problems through abstraction and refinement.

3. PROBLEM STATEMENT

In this paper, we focus on discrete linear time varying (LTV) systems. Consider the discrete type linear control system evolving according to the equation:

$$x_{t+1} = A_t x_t + B_t u_t + C_t a_t, \tag{1}$$

where for each time instant $t \in \mathbb{N}$, $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state vector of the controlled plant, $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$ is controller input to the plant, and $a_t \in \mathcal{A} \subseteq \mathbb{R}^l$ is adversarial input to the plant. For a fixed time horizon $T \in \mathbb{N}$, let us denote sequences of controller and adversary inputs by $\mathbf{u} \in \mathcal{U}^T$ and $\mathbf{a} \in \mathcal{A}^T$. In addition to the sequence of matrices A_t, B_t, C_t, and a time bound T, the *linear adversarial reach-avoid control* problem or *ARAC* in short is parameterized by: (i) three sets of states $Init, Safe, Goal \subseteq \mathcal{X}$ called the *initial, safe* and *goal* states, (ii) a set $Ctr \subseteq \mathcal{U}^T$ called the *controller constraints*, and (iii) a set $Adv \subseteq \mathcal{A}^T$ called the *adversary constraints*. We will assume finite representations of these sets such as polytopes and we will state these representational assumptions explicitly later. A controller input sequence $\mathbf{u}$ is *admissible* if it meets the constraints Ctr, that is, $\mathbf{u} \in Ctr$, and a adversarial input sequence is *admissible* if $\mathbf{a} \in Adv$. We define what is means to solve a *ARAC* problem with an open loop controller strategy.

Definition 1. *A* solution *to a ARAC is an input sequence* $\mathbf{u} \in Ctr$ *such that for any initial state* $x \in Init$ *and any admissible sequence of adversarial inputs* $\mathbf{a} \in Adv$*, the states visited by the system satisfies the condition:*

- (Safe) *for all* $t \in \{0, \ldots, T\}$, $x_t \in Safe$ *and*
- (Winning) $x_T \in Goal$.

In this paper we propose an algorithm that given a *ARAC* problem, either computes its solution or proves that there is none. In the next section, we discuss how the problem captures instances of control synthesis problems for cyber-physical systems under several different types of attacks.

Helicopter Autopilot Example

To make this discussion concrete we consider an autonomous helicopter. The state vector of the plant $x \in \mathbb{R}^{16}$; the control input vector $u \in \mathbb{R}^4$ with bounded range of each component. The descriptions of the state and input vectors are in Table 1. The dynamics of the helicopter is given in [19], which can be discretized into a linear time-invariant system: $x_{t+1} = Ax_t + Bu_t$. The auto-pilot is supposed to take the helicopter to a waypoint in a 3D-maze within a bounded time T (*Goal*) and avoid the mapped building and trees. The complement of these obstacles in the 3D space define the *Safe* set (see Figure 1).

The computation of the control inputs (u_t) typically involves sensing the observable part of the states, computing the inputs to the plant, and feeding the inputs through actuators. In a cyber-physical system, the mechanisms involved in each of these steps can be attacked and different attacks give rise to different instances of *ARAC*.

Controller and Actuator attacks. An adversary with software privileges may compromise a part of the controller software. A network-level adversary may inject spurious packets

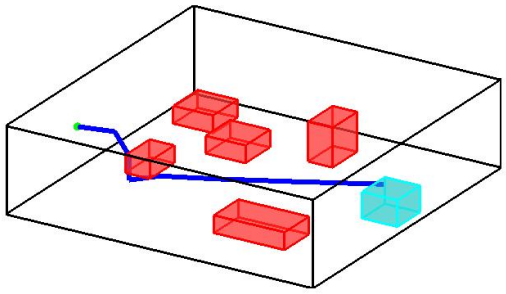

Figure 1: Helicopter fly through scene. Red boxes are the obstacles, the cyan box on the right is the goal states, the green ball on the left is a set of initial states and the blue curve is a sampled trajectory of the helicopter with a random adversary input.

in the channel between the controller and the actuator. An adversary with hardware access may directly tamper with the actuator and add an input signal of a_t. Under many circumstances, it is reasonable to expect these attacks to be transient or short-lived compared T (for example, otherwise they will be diagnosed and mitigated). Then the actual input to the system becomes $u'_t = u_t + a_t$ and the dynamics of the complete system is modified to $x_{t+1} = Ax_t + Bu_t + Ba_t$, which gives an instance of $ARAC$.

Sensor attacks. Another type of adversary spoofs the helicopter's sensors, the GPS, the gyroscope, so that the position estimator is noisy. Consider a control systems where the adversary-free control u_t is a function on the sequence of sensor data. If the adversary injects an additive error to the sensors, then the control inputs computed based on this inaccurate data will be added an error; also the initial state will have uncertainty. We model the additive error by the adversary input a_t. Once again, this gives rise to an instance of $ARAC$. Assuming that the injection of a_t requires energy and that the adversary has limited energy for launching the attack then gives rise the adversary class $Adv = \sum_{i=0}^{T} ||a_t||^2 \leq b$ where b is the energy budget.

States/ Inputs	Description
$[p_x, p_y, p_z]$	Cartesian Coordinates
$[u, v, w]$	Cartesian Velocities
$[p, q, r]$	Euler Angular Rates
$[a, b, c, d]$	Flapping Angles
$[\varphi, \phi, \theta]$	Euler Angles
u_z	Lateral Cyclic Deflection in [-1,1]
u_x	Longitudinal Cyclic Deflection in [-1,1]
u_p	Pedal Control Input in [-1,1]
u_c	Collective Control Input in [0,1]

Table 1: States and inputs of the helicopter model.

4. ALGORITHM FOR LINEAR $ARAC$

4.1 Preliminaries and Notations

For a natural number $n \in \mathbb{N}$, $[n]$ is the set $\{0, 1, \ldots, n-1\}$. For a sequence A of objects of any type with n elements, we refer to the i^{th} element, $i \leq n$ by A_i. For a real-valued vector $v \in \mathbb{R}^n$, $||v||$ is its ℓ^2-norm. For $\delta \geq 0$, the set $\mathcal{B}_\delta(v)$ denotes the closed ball $\{x \in \mathbb{R}^n \mid ||v - x|| \leq \delta\}$ centered at v. For a parameter $\epsilon > 0$ and a compact set $A \subseteq \mathbb{R}^n$, an ϵ-cover of A is a finite set $C = \{a_i\}_{i \in I} \subseteq A$ such that $\cup_{i \in I} \mathcal{B}_\epsilon(a_i) \supseteq A$. For two sets $A, B \subseteq \mathbb{R}^n$, the *direct sum* $A \oplus B = \{x \in \mathbb{R}^n : \exists a \in A, \exists b \in B, a + b = x\}$. For a vector v, we denote $A \oplus v$ as $A \oplus \{v\}$. Sets in $\mathbb{R}^n$ will be represented by finite union of balls or polytopes. An *n-dimensional polytope* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is specified by a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, where m is the number of constraints. A *polytopic set* is a finite union of polytopes and is specified by a sequence of matrices and vectors. A polytopic set can be written in Conjunctive Normal Form (CNF), where (i) the complete formula is a conjunction of *clauses*, and (ii) each clauses is disjunction of linear inequalities.

In this paper, we will assume that the initial set $Init$ is given as a ball $\mathcal{B}_\delta(\theta) \subseteq \mathcal{X}$ for some $\theta \in \mathcal{X}$ and $\delta > 0$. We also fix the time horizon T. The set Adv is specified by a *budget* $b \geq 0$: $Adv = \{\mathbf{a} \in \mathcal{A}^T : \sum_t ||a_t||^2 \leq b\}$. The set Ctr is specified by a polytopic set.

For a sequence of matrices $\{A_t\}_{t \in \mathbb{N}}$, for any $0 \leq t_0 < t_1$, we denote the *transition matrix* from t_0 to t_1 inductively as $\alpha(t_1, t_0) = A_{t_1 - 1}\alpha(t_1 - 1, t_0)$ and $\alpha(t_0, t_0) = I$.

A *trajectory of length* T for the system is a sequence $x_0, x_1, \ldots, x_T$ such that $x_0 \in Init$ and each x_{t+1} is inductively obtained from Equation (1) by the application of some admissable controller and adversary inputs. The t^{th} state of a trajectory is uniquely defined by the choice of an initial state $x_0 \in Init$, an admissible control input $\mathbf{u} \in Ctr$ and an admissible adversary input $\mathbf{a} \in Adv$. We denote this state as $\xi(x_0, \mathbf{u}, \mathbf{a}, t)$.

The notion of a trajectory is naturally extended to sets of trajectories with sets of initial states and inputs. For a time $t \in [T + 1]$, a subset of initial states $\Theta \subseteq Init$, a subset of adversary inputs $\mathbf{A} \subseteq Adv$, and a subset of controller inputs $\mathbf{U} \subseteq Ctr$, we define:

$$\mathsf{Reach}(\Theta, \mathbf{U}, \mathbf{A}, t) = \{\xi(x_0, \mathbf{u}, \mathbf{a}, t) : x_0 \in \Theta \ \wedge \ \mathbf{a} \in \mathbf{A}\}.$$

For a singleton $\mathbf{u} \in \mathcal{U}$, we write $\mathsf{Reach}(\Theta, \{\mathbf{u}\}, Adv, t)$ as $\mathsf{Reach}(\Theta, \mathbf{u}, t)$. To solve $ARAC$ then we have to decide if

$$\begin{aligned} \exists\ \mathbf{u} \in Ctr\ : (\wedge_{t \in [T+1]} \mathsf{Reach}(Init, \mathbf{u}, t) \subseteq Safe) \\ \wedge\ \mathsf{Reach}(Init, \mathbf{u}, T) \subseteq Goal. \end{aligned} \tag{2}$$

This representation hides the dependence of the Reach sets on the set of adversary choices.

4.2 Decoupling

In this section, we present a technique to decouple the $ARAC$ problem. The decomposition relies on a result from robust control that enables us to precisely compute the reachable states of the system in terms of a symbolic simulation of the adversary-free dynamics and the total uncertainty induced by the adversary. In Section 4.6, we present an algorithm that performs this decomposition such as to eliminate the universal quantifier on the adversary's choices and initial states in Definition 2 and 3.

4.3 Adversarial Leverage

Definition 2. *For any $t \in [T+1]$, the* adversary leverage *at t, initial state $x_0 \in Init$, and any control $\mathbf{u} \in Ctr$, the* adversary leverage *is a set $R(x_0, \mathbf{u}, t)$ such that*

$$\mathsf{Reach}(x_0, \mathbf{u}, t) = \xi(x_0, \mathbf{u}, 0, t) \oplus R(x_0, \mathbf{u}, t) \tag{3}$$

Informally, the adversary leverage captures how much an adversary can drive the trajectory from an adversary-free trajectory.It decomposes the reach set $\mathsf{Reach}(x_0, \mathbf{u}, t)$ into two parts: a deterministic adversary-free trajectory $\xi(x_0, \mathbf{u}, 0)$, and the reachtube $R(x_0, \mathbf{u}, t)$ that captures the nondeterminism introduced by the adversary. Our solution for $ARAC$ heavily relies on computing over-approximations of reach sets and to that end, observe that is suffices to over-approximate adversary leverage. For certain classes of non-linear systems, it can be over-approximated statically using techniques from robust control, such as H_∞ control. It can also be approximated dynamically by reachability algorithms that handle nondeterministic modes (see, for example [20, 21]).

For the $ARAC$ problem with linear dynamics described in (1), where the adversary input $Adv = \{\mathbf{a} \in \mathcal{A}^T : \sum_t ||a_t||^2 \leq b\}$ is defined by a budget $b \geq 0$, we can compute adversary leverage precisely. The following lemma is completely standard in linear control theory.

Lemma 1. *For any time $t \in [T+1]$, if the controllability Gramian of the adversary $W_t = \sum_{s=0}^{t-1} \alpha(t, s+1) C_s C_s^T \alpha^T(t, s+1)$ is invertible, then*

$$R(x_0, \mathbf{u}, t) = \{x \in \mathbb{R}^n : x^T W_t^{-1} x \leq b\}$$

is the precise adversary leverage at t.

Proof. For $t \in [T+1]$, we have

$$x_t = \alpha(t,0)x_0 + \sum_{s=0}^{t-1} \alpha(t, s+1) B_s u_s + \sum_{s=0}^{t-1} \alpha(t, s+1) C_s a_s. \tag{4}$$

Since $\xi(x_0, \mathbf{u}, 0, t) = A^t x_0 + \sum_{s=0}^{t-1} \alpha(t, s+1) B_s u_s$, we have

$$R(x_0, \mathbf{u}, t) = \{x \in \mathbb{R}^n : x = \sum_{s=0}^{t-1} \alpha(t, s+1) C_s a_s \wedge \sum_{t=0}^{T-1} ||a_s||^2 \leq b\},$$

which is the set $\{x \in \mathbb{R}^n : x^T W_t^{-1} x \leq b\}$, with controllability Gramian W_t. □

The above lemma establishes a precise adversary leverage as an ellipsoid defined by the controllability Gramian W_t and b. In this case, the ellipsoid is independent of x_0 an $\mathbf{u}$ and only depends on t. Here on, we will drop the arguments of R when they are reduandant or clear from context. If W_t is singular for some $t \in [T+1]$, then replace the inverse of W_t by its pseudo-inverse and the set R is an ellipsoid in the controllable subspace.

4.4 Uncertainty in Initial Set

Following the above discussion, we show that a similar decomposition of the reachable states is possible with respect to the uncertainty in the initial state.

Definition 3. *Consider the initial set $Init$ to be $B_\delta(x_0)$ for some $\delta > 0$ and $x_0 \in \mathcal{X}$. For a $t \in [T+1]$ and a control input $\mathbf{u}$, the* initialization factor *at t is a set $B(x_0, \mathbf{u}, t)$, such that*

$$\mathsf{Reach}(B_\delta(x_0), \mathbf{u}, 0, t) = \xi(x_0, \mathbf{u}, 0, t) \oplus B(x_0, \mathbf{u}, t). \tag{5}$$

The initialization factor captures the degree to which the uncertainty δ in the initial set can make the adversary-free trajectories deviate. For general nonlinear models, we will have to rely on over-approximating initialization factor [], but for the liner version of $ARAC$ the following lemma provides a precise procedure for computing it.

Lemma 2. *For an initial set $Init = \mathcal{B}_\delta(\theta) \subseteq \mathbb{R}^n$, for any $t \in [T+1]$, input $\mathbf{u} \in Ctr$, if the matrix $\alpha(t,0)^T \alpha(t,0)$ is invertible then*

$$B(\theta, \mathbf{u}, t) = \{x \in \mathbb{R}^n \ : \ x^T [\alpha^T(t,0)\alpha(t,0)]^{-1} x \leq \delta^{1/2}\}$$

is the precise initialization factor at t.

If the matrix A is singular, then a similar statement holds in terms of the pseudo-inverse of $[\alpha^T(t,0)\alpha(t,0)]$. Thus, initialization factor is an ellipsoid defined by A, t and δ and is independent of x_0 an $\mathbf{u}$. We will drop the arguments of B when they are redundant or clear from context.

4.5 Adversary-free Constraints

Using the decomposition of the reach set given by the above lemmas, we will first solve a new reach-avoid synthesis problem for the adversary-free system. To construct this new problem we will modify the safety and winning constraints of the $ARAC$. For a given time instant, the new constraints are obtained using the same approach as in robotic planning with The synthesis problem requires a solution to a sequence of such problems.

Definition 4. *Given a set $S \subseteq \mathbb{R}^n$ and a compact convex set $R \subseteq \mathbb{R}^n$, a set $S' \subseteq \mathbb{R}^n$ is a* strengthening of S by R *if*

$$S' \oplus R \subseteq S. \tag{6}$$

A strengthening S' is *precise* if it equals $R \ominus S$. The strengthening S' is a subset of S that is shrunk by the set R. If S is a polytopic set and R is a convex compact set then exact solutions to the following optimization problem yields precise strengthening.

Lemma 3. *For a half hyperplane $S = \{x \in \mathbb{R}^n : c^T x \leq b\}$ and a convex compact set R, a precise strengthening of S by R is $S' = \{x \in \mathbb{R}^n : c^T x \leq b - c^T x^*\}$ such that*

$$x^* = \arg\min_{x \in R} -c^T x. \tag{7}$$

.

Proof. Fix any $x \in R$ and $y \in S'$. From the definition of S', $c^T y + b^* \leq b$. Since x^* minimizes $-c^T x$ in R and $x \in R$, we have $-c^T x \geq -c^T x^* = b^*$. It follows that $c^T(x+y) \leq c^T y + c^T x^* \leq c^T y + b^* \leq b$. Thus $x + y \in S$ and therefore $S' \oplus R \subseteq S$.

For any $y \in S$, it holds that $c^T y \leq b$. Let $y' = y - x^*$. It follows that $c^T y' = c^T y - c^T x^* \leq b - c^T x^*$. Thus $y' \in S'$. Combined with $x^* \in R$, $y = y' + x^* \in S' \oplus R$. Therefore $S' \oplus R \subseteq S$. □

Since a polytopic set is a union of intersections of linear inequalities, the above lemma generalizes to polytopic sets in natural way.

Corollary 4. *For a polytopic set $S = \{x \in \mathbb{R}^n \ : \ \vee_{i \in [m]} A_i x \leq b_i\}$ and a compact convex set $R \subseteq \mathbb{R}^n$,*

$$S' = \{x \in \mathbb{R}^n \ : \ \bigvee_{i \in [m]} A_i x \leq b_i - b_i^*\},$$

is a precise strengthening of S by R. Here the j^{th} element of b_i^* equals $c^T x^*$ with c^T being the j^{th} row of A_i and x^* is the solution of (7).

4.6 An Algorithm for Linear *ARAC*

We present algorithm 1 for solving the linear version of the *ARAC* problem.

Algorithm 1: $Synthesis(Init, Safe, Goal, Adv, Ctr, T)$

```
for t ∈ [T+1] do
    R_t ← AdvDrift(Adv, t);
    B_t ← InitCover(Init, t);
    Safe'_t ← Strengthen(Safe, R_t, B_t);
end
Goal' ← Strengthen(Goal, R_T, B_T);
(u, Failed) ← SolveSMT(θ, Safe', Goal', Ctr, T);
return (u, Failed)
```

The subroutine *AdvDrift* computes a precise adversary leverage R_t for every time $t \in [T+1]$. From Lemma 1, R_t is an ellipsoid represented by the controllability Gramian and the constant b. The subroutine *InitCover* computes a initialization factor described in Lemma 2 for each t. The subroutine *Strengthen* computes a precise strengthening of the safety constraints *Safe* by both sets R_t and B_t. From Corollary 4, the strengthening is computed by solving a sequence of optimization problems. Since R_t and B_t are both ellipsoids (Lemma 1 and 2), the optimization problems solved by *Strengthen* are quadratically constrained linear optimization problems and are solved efficiently by second-order cone programming [22] or semidefinite programming [23]. For each $t \in [T+1]$, the set *Safe* is strengthened by the corresponding adversary drift R_t to get $Safe'_t$. The *Goal* set is strengthened respect to the adversary drift at the final time T to get $Goal'$. Finally, *SolveSMT* makes a call to an SMT solver to check if there exists a satisfiable assignment $\mathbf{u} \in Ctr$ for quantifier-free formula (8):

$$\exists\, \mathbf{u} \in Ctr \wedge (\wedge_{t\in[T+1]} \xi(\theta, \mathbf{u}, 0, t) \in Safe'_t) \wedge \xi(\theta, \mathbf{u}, 0, T) \in Goal'. \tag{8}$$

For the class of problems we generate, the SMT solver terminates and either returns a satisfying assignment $\mathbf{u}$ or it proclaims the problem is unsatisfiable by returning *Failed*. If *AdvDrift*, *InitCover* and *Strengthen* compute adversary leverage, initialization factor and strengthening precisely, then Algorithm 1 is a sound and complete for the linear *ARAC* problem.

Theorem 5. *Algorithm 1 outputs $\mathbf{u} \in Ctr$ if and only if $\mathbf{u}$ solves ARAC.*

Proof. Suppose Algorithm returns $\mathbf{u} \in Ctr$. We will first show that $\mathbf{u}$ solves *ARAC*. Since $\mathbf{u}$ satisfies constraints (8), for every $t \in [T+1]$, $\xi(\theta, \mathbf{u}, 0, t) \in S_t$. Since S_t is a strengthening of *Safe* by R_t and B_t, we have $S_t \oplus R_t \oplus B_t \subseteq Safe$. Thus,

$$\xi(\theta, \mathbf{u}, 0, t) \oplus S_t \oplus B_t \subseteq Safe. \tag{9}$$

By Definition 2 and 3, we have

$$\begin{aligned} \xi(\theta, \mathbf{u}, 0, t) \oplus Safe'_t \oplus B_t &\supseteq \mathsf{Reach}(\theta, \mathbf{u}, Adv, t) \oplus B_t \\ &\supseteq \mathsf{Reach}(Init, \mathbf{u}, t). \end{aligned} \tag{10}$$

Combining (9) and (10), we have $\mathsf{Reach}(Init, \mathbf{u}, t) \subseteq Safe$. That is the safety condition of (2) holds. Similarly, since $Goal'$ is the strengthening of *Goal* by R_T and B_T, we have $\mathsf{Reach}(Init, \mathbf{u}, T) \subseteq Goal$. The winning condition also holds.

On the other side, suppose $\mathbf{u} \in Ctr$ solves *ARAC*, it satisfies (2). Since the adversary leverage R_t, initialization factor B_t and strengthening $Safe'$, $Goal'$ are computed precisely, Equations (9) and (10) take equality. Thus, for any $t \in [T+1]$, $\xi(\theta, \mathbf{u}, 0, t) \in Safe'_t$ and $\xi(\theta, \mathbf{u}, 0, T) \in Goal'$. Therefore $\mathbf{u}$ is returned by Algorithm 1. □

The completeness of the algorithm is based on two facts: (i) adversary leverage, initialization factor and strengthening can be computed precisely, and (ii) the SMT solver is complete for formula (8). The exact computation of adversary leverage and initialization factor require that the initial state *Init* and admissible adversary *Adv* are described by ℓ^2 balls. Since *Ctr*, *Safe'* and *Goal'* are polytopic sets, formula (8) is a quantifier-free theory in linear arithmetic, which can be solved efficiently for example by algorithm DPLL(T) [24].

5. GENERALIZATIONS

In this section, we discuss two orthogonal generalizations of linear *ARAC* and algorithms for solving them building on the algorithm *Synthesis*. First in Section 5.1, we present an approximate approach to solve a problem where *Init*, *Adv* and *Ctr* are general compact convex sets. Then, in Section 5.2, we modified the definition of linear *ARAC* problem such that the controller can be a function of the initial states. A solution of this problem is a look-up table, where the controller choose a sequence of open loop control depending on the initial state.

5.1 Synthesis for Generalized Sets

We generalize the linear *ARAC* problem described in Section 4.1 such that $Init \subseteq \mathcal{X}$, $Ctr \subseteq \mathcal{U}^T$ and $Adv \subseteq \mathcal{A}^T$ are assumed to be some compact subsets of Euclidean space. For a precision parameter $\epsilon > 0$, the generalized *ARAC* problem can be approximated by a linear *ARAC* problem. We define robustness of a *ARAC* problem.

We present an extension of *Synthesis* to solve this problem. For a parameter $\epsilon > 0$, and compact convex sets $Init, Adc, Ctr$, we construct a tuple $(\Theta, \mathbf{A}, \mathcal{C})$ such that

(i) $\Theta = \{\theta_i\}_{i\in I}$ is an ϵ-cover of initial set *Init*, that is, $Init \subseteq \cup_i B_\epsilon(\theta_i)$.

(ii) $\mathbf{A} = \{\mathbf{a_j}\}_{\mathbf{j}\in\mathbf{J}}$ is an ϵ-cover of the adversary. Here each $\mathbf{a}_j$ is seen as a vector in Euclidean space $\mathcal{A}^T$ and the union of ϵ-balls around each $\mathbf{a}_j$ over-approximates *Adv*.

(iii) $\mathcal{C} \subseteq Ctr \subseteq \mathcal{U}^T$ is a polytopic set such that $d_H(\mathcal{C}, Ctr) \leq \epsilon$, That is, $\mathcal{C}$ under-approximates the actual constraints of control *Ctr*, with error bounded by ϵ measured by Hausdorff distance.

The modified algorithm to approximately solve the generalized *ARAC* problem follows the same steps as Algorithm 1

from line 1 to line 6. The only change is in line 7, where instead of solving an SMT formula (8) we solve (11).

$$\exists\ \mathbf{u}\ :\ \mathbf{u}\in\mathcal{C}\ \wedge$$
$$(\wedge_{t\in[T+1]}\wedge_{i\in I}\wedge_{j\in J}\xi(\theta_i,\mathbf{u},\mathbf{a}_j,t)\in \mathit{Safe}'_t)\ \wedge \tag{11}$$
$$(\wedge_{i\in I}\wedge_{j\in J}\xi(\theta,\mathbf{u}_i,\mathbf{a}_j,T)\in \mathit{Goal}')$$

The soundness of this modified algorithm is independent of the choice of $\epsilon > 0$. That is, if it returns a satisfiable assignment $\mathbf{u}$, then $\mathbf{u}$ solves the $ARAC$ problem.

Lemma 6. *If the modified algorithm returns $\mathbf{u}\in\mathcal{C}$, then $\mathbf{u}$ solves linear generalized $ARAC$.*

Proof. Suppose $\mathbf{u}\in\mathcal{C}\subseteq Ctr$ satisfies (11). Since Θ and $\mathbf{A}$ are ϵ-cover of $Init$ and Adv, there exist a initial state $\theta_i\in$ for any $t\in[T+1]$ we have

$$\mathsf{Reach}(\mathit{Init},\mathbf{u},\mathit{Adv},t)\subseteq\mathsf{Reach}(\cup_{i\in I}\mathcal{B}_\epsilon(\theta_i),\mathbf{u},\cup_{j\in J}\mathcal{B}_\epsilon(\mathbf{a}_j),t).$$

Let R_t and B_t be the precise adversary leverage and initialization factor as in Algorithm 1. From Lemma 1 and 2, R_t and B_t are independent on the initial state and adversary input. Therefore,

$$\begin{aligned}&\mathsf{Reach}(\mathit{Init},\mathbf{u},\mathit{Adv},t)\\ =\ &\cup_{i\in I}\cup_{j\in J}\mathsf{Reach}(\mathcal{B}_\epsilon(\theta_i),\mathbf{u},\mathcal{B}_\epsilon(\mathbf{a}_j),t)\\ =\ &\cup_{i\in I}\cup_{j\in J}(\xi(\theta_i,\mathbf{u},\mathbf{a}_j,t)\oplus R_t\oplus B_t)\\ =\ &(\cup_{i\in I}\cup_{j\in J}\xi(\theta_i,\mathbf{u},\mathbf{a}_j,t))\oplus R_t\oplus B_t.\end{aligned}\tag{12}$$

From formula (11) implies that $(\cup_{i\in I}\cup_{j\in J}\xi(\theta_i,\mathbf{u},\mathbf{a}_j,t))\subseteq \mathit{Safe}'_t$ for any $t\in[T+1]$ and $(\cup_{i\in I}\cup_{j\in J}\xi(\theta_i,\mathbf{u},\mathbf{a}_j,T))\subseteq \mathit{Goal}'$. Since Safe'_t is an $R_t\oplus B_t$ strengthening of Safe, it follows from Definition 4 and (12) that $\mathsf{Reach}(\mathit{Init},\mathbf{u},\mathit{Adv},t)\subseteq \mathit{Safe}$ for all $t\in[T+1]$ and $\mathsf{Reach}(\mathit{Init},\mathbf{u},\mathit{Adv},T)\subseteq \mathit{Goal}$. That is, $\mathbf{u}$ solves the generalized linear $ARAC$. □

We observe that if the approximated algorithm successfully synthesize a control, the control solves the generalized linear $ARAC$ problem, no matter what value $\epsilon > 0$ takes. Moreover, as the parameter ϵ converges to 0, we have $\cup_{i\in I}\mathcal{B}_\epsilon(\theta_i)$, $\cup_{j\in J}\mathcal{B}_\epsilon(\mathbf{a}_j)$ and $\mathcal{C}$ converge to the exact $Init$, Adv and Ctr, respectively.

5.2 State-dependent Control

In this section, we keep the same definition of $Init$, Adv and Ctr as in Section 4.1, however, we consider a variant of $ARAC$ that allows the choice of control $\mathbf{u}$ to be depend on the initial state of the system. That is, we have to decide if

$$\forall\ x_0\in\mathit{Init}\ :\ \exists\ \mathbf{u}\in\mathit{Ctr}\ :$$
$$(\wedge_{t\in[T+1]}\mathsf{Reach}(x_0,\mathbf{u},t)\subseteq\mathit{Safe})\wedge\ \mathsf{Reach}(x_0,\mathbf{u},T)\subseteq\mathit{Goal}. \tag{13}$$

A solution to this generalized $ARAC$ problem is a look-up table $\{(\mathcal{I}_i,\mathbf{u}_i)\}_{i\in I}$ such that (i) the union $\cup_{i\in I}\mathcal{I}_i\supseteq\mathit{Init}$ covers the initial set, and (ii) for every $x_0\in\mathcal{I}_i$, $\mathbf{u}_i$ is an admissible input such that the constraints in (13) hold.

We present an Algorithm 2 to solve this problem and it uses *Synthesis* as an subroutine. If the algorithm succeeds, it returns a look-up table Tab which solves the above state-dependent variant of $ARAC$.

The parameters $Adv, Ctr, Safe, Goal, T$ are invariant in the algorithm, thus we omit it as arguments of *Synthesis*. The variable ϵ is initialized as the diameter of the initial set $Init$ (line 1). The subroutine Cover$(Init,\epsilon)$ in line first computes an ϵ-cover $\{\theta_i\}_{i\in I}$ of $Init$, and then append each θ_i with the parameter ϵ. The set $\mathcal{S}$ stores all such pairs (θ,ϵ), such that the ϵ-ball around θ is yet to examined by the algorithm for *Synthesis*. For each ball $\mathcal{B}_\epsilon(\theta)$ in $\mathcal{S}$, the subroutine *Synthesis* is possibly called twice for both the ball $\mathcal{B}_\epsilon(\theta)$ and the single initial state θ to decide whether the *Synthesis* is successful, a failure, or whether further refinement is needed.

Algorithm 2: *TableSynthesis*

```
1  ε ←Dia(Init);
2  S ←Cover(Init, ε);
3  Tab← ∅;
4  while S ≠ ∅ For (θ, ε) ∈ S do
5  |  S ← S/{(θ, ε)};
6  |  if Synthesis(B_ε(θ)) returns u ∈ Ctr then
7  |  |  Tab← Tab∪{(B_ε(θ), u)};
8  |  else if Synthesis(B_ε(θ)) failed then
9  |  |  return (θ,Failed)
10 |  else
11 |  |  S ← S ∪ Cover(Init ∩ B_ε(θ), ε/2);
12 |  end
13 end
14 return (Tab, Success)
```

Theorem 7. *If* TableSynthesis *returns* (Tab,Success), *then* Tab *solves the state-dependent $ARAC$. Otherwise if* Tablesynthesis *returns* (θ,Failed), *then there is no solution for initial state θ.*

Proof. We first state an invariant of the while loop which can be proved straightforwardly through induction. For any iteration, suppose Tab$=\{(\mathcal{B}_{\epsilon_i}(\theta_i),\mathbf{u}_i)\}_{i\in I}$ and $\mathcal{S}=\{\theta'_j,\epsilon'_j\}_{j\in J}$ are the valuations of Tab and $\mathcal{S}$ at the beginning of the iteration. Then we have $(\cup_{i\in i}\mathcal{B}_{\epsilon_i}(\theta_i))\cup(\cup_{j\in J}\mathcal{B}_{\epsilon'_j}(\theta'_j))\supseteq\mathit{Init}$.

Suppose $TableSynthesis$ returns (Tab,Success) with Tab$=\{(\mathcal{B}_{\epsilon_i}(\theta_i),\mathbf{u}_i)\}_{i\in I}$. From line 4, $\mathcal{S}=\varnothing$. From the loop invariant, we have $\cup_{i\in i}\mathcal{B}_{\epsilon_i}(\theta_i)\supseteq\mathit{Init}$. Moreover for any $(\mathcal{B}_\epsilon(\theta),\mathbf{u})\in$Tab, from line 6 and Theorem 5, for any $x_0\in\mathcal{B}_\epsilon(\theta)$, $\mathbf{u}$ is an admissible input such that constraints in **??** hold. Thus Tab solves the state-dependent $ARAC$.

Otherwise suppose $TableSynthesis$ returns (θ,Failed). From line 8 and Theorem 5, there is no admissible $\mathbf{u}$ solve the $ARAC$ from θ. □

The Algorithm 2 is sound, that is, if the algorithm terminates, it always returns the right answer. For general sets of Adv and Ctr the approach from Section 5.1 can be combined Algorithm 2 to get state dependent (but $\mathbf{u}$ and $\mathbf{a}$ oblivious) controllers.

6. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

We have implemented the algorithm *Synthesis* in a prototype tool in Python. The optimization problem presented in Lemma 3 is solved by a second-order cone programming solver provided by package CVXOPT [9]. The quantifier-free SMT formula (8) is solved by Z3 solver [10]. In Section 6.1 and 6.2, we present the implementation of the basic

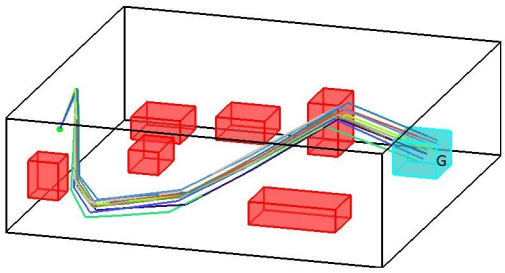

Figure 2: Sampled Trajectories of Helicopter Auto-pilot. Safety and winning conditions hold.

algorithm *synthesis*, show an example in detail, present the experiment results and discuss the complexity of the algorithm. In Section 6.3 and 6.4, we present several different applications of *Synthesis*.

6.1 Synthesizing Adversary Resistant Controllers

We have solved several linear *ARAC* problems for a 16-dimensional helicopter system (as described in 3) and a 4-dimensional vehicle.

We illustrate an instance of the synthesis of the helicopter auto-pilot for time bound $T = 9$ in Figure 2. The state variables, control input variables and the constraint Ctr of the system are listed in Table 1. We model an actuator intrusion attack such that the control input is tempered by an amount of a_t at each time $t \in [T]$. The total amount of spoofing is bounded by a budget $b = 1$.

A control $\mathbf{u} = \{u_t\}_{t\in[T]}$ is synthesized by *Synthesis*. We randomly sample adversary inputs $\mathbf{a}$ with $\sum_{t\in[T]} ||a_t||^2 = b$, and visualize the corresponding trajectories with control $\mathbf{u}$ in Figure 2.

Besides the Helicopter model, we studied an discrete variation of the navigation problem of a 4-dimensional vehicle, where the states are positions and velocities in Cartesian coordinates, and the controller and adversary compete to decide accelerations in both direction.

The experimental results for different instances are listed in Table 2, where the columns represent (i) the model of the complete system, (ii) the dimension of state, control input and adversary input vectors, (iii) the time bound, (iv) the length of formula representing *Safe* and number of obstacles, (v) the length of formula representing *Goal* and Ctr, (vi) the length of the quantifier-free formula in (2), (vii) the synthesis result, and (vii) the running time of the synthesis algorithm.

From the result, we observe that the algorithm can synthesize controller for lower dimensional system for a relatively long horizon (320) for reasonable amount of time. For higher dimensional system (16-dimensional), the approach scales to an horizon $T = 15$. The run time of the algorithm grows exponentially with the time bound T. By Comparing row 2-4, we observe that the runtime grows linearly with the number of obstacles.

6.2 Discussion on Complexity of Safety Constraints

Let the quantifier-free constraints in (2) be specified by an CNF formula ϕ, where each atomic proposition is a linear constrain. We denote $|\phi|$ as the length of the CNF formula which is the number of atomic propositions in ϕ. Notice that if we convert an CNF formula into a form of union of polytopes, the size of the formula can grow exponentially. Similarly, let CNF formula ϕ_{Safe}, ϕ_{Goal} and ϕ_{Ctr} specify the constraints *Safe*, *Goal* $\subseteq \mathcal{X}$ and $Ctr \subseteq \mathcal{U}^T$. It can be derived from (2) that $|\phi| = T|\phi_{Safe}| + |\phi_{Goal}| + |\phi_{Ctr}|$. If fixed the length of the projection of ϕ_{Ctr} on control u_t for each t, that is, we assume the controller constraints at different times are comparably complex, then $|\phi_{Ctr}|$ grows linear with the time bound T. Suppose the length of $|\phi_{Safe}|, |\phi_{Goal}|$ are constant, then the length of ϕ is linear to the time bound T.

The length of ϕ_{Safe} is a function of the number and complexity of obstacles. Suppose that the safe region *Safe*$'$ is obtained by adding an polytopic obstacle $O = \{x \in \mathbb{R}^n : Ax < b\}$ to a safe region *Safe*. One measure of complexity of the obstacle is the number of rows of the matrix A. Then, the resulting safe region is *Safe*$' =$ *Safe*$\backslash O$, which implies

$$\phi_{Safe'} = \phi_{Safe} \wedge \neg(Ax < b) = \phi_{Safe} \wedge (\vee_i - A_i x \leq -b),$$

where A_i is the i^{th} row of A. Therefore the length of ϕ_{Safe} increases linearly with the number of obstacles and the number of faces in every obstacle.

In the experiments, we observe that the running time of Z3 to solve the SMT formula varies on a case by case basis. The size of obstacles, the volume of the obstacle-free region and the length of significant digits of entries the constraints and dynamic matrices also affect the running time.

6.3 Vulnerability Analysis of Initial States

Using *Synthesis*, we can examine the vulnerability of initial states to attackers. Fixing a controller constraint Ctr, a time bound T, safety condition *Safe* and winning condition *Goal*, for each initial state *Init*, there exists a maximum critical budget b_{mfc} of the adversary *Adv*, such that beyond this budget, the problem becomes infeasible. The lower the b_{mfc} for an initial state is, it is vulnerable to a weaker adversary. The maximum budget can be found by a binary search on the adversary budget with *Synthesis*.

We examine the vulnerability of an instance of the 4-dimensional autonomous vehicle system. The result is illustrated in Figure 3, where the box at the bottom represent the *Goal*, the red regions represent the obstacle whose complement is the *Safe*, the green-black on the top region is the *Init*. The black regions are most vulnerable with $b_{mfc} = 0$ and the lightest green region are least vulnerable with $b_{mcc} = 1.8$. We see that the region closer to an obstacle are darker as an adversary with relatively small budget (b) can make the vehicle run into an obstacle. We also observe that the dark regions are shifted towards the center since the obstacles are aggregated at the center of the plane. Avoiding them may cause a controller run out of the time bound.

6.4 Attack Synthesis

The *Synthesis* subroutine can also be used to generate attacks by swathing the roles of the adversary and the controller. In this section, we synthesize adversarial attacks to the 4-dimensional vehicle such that the system will be driven to unsafe states in a bounded time T. That is, for a state $x \in \mathcal{X}$, we decide whether

$$\begin{aligned} &\exists\ \mathbf{a} \in Adv\ \forall\ \mathbf{u} \in Ctr : \\ &(\vee_{t\in[T]} \xi(x, \mathbf{u}, \mathbf{a}, t) \in Unsafe). \end{aligned} \tag{14}$$

Complete System	# x, u, a	T	$\lvert\phi_{Safe}\rvert, \#Obs$	$\lvert\phi_{Goal}\rvert, \lvert\phi_{Ctr}\rvert$	$\lvert\phi\rvert$	Result	Run Time (s)
Vehicle	4,2,2	40	16, 3	4, 160	804	unsat	2.79
		80	20, 4	4, 320	1924	sat	16.49
		80	44, 10	4, 320	3844	sat	35.22
		80	84, 20	4, 320	7044	sat	53.8
		160	20, 5	4, 640	3844	sat	91.78
		320	24, 6	4, 1280	8964	sat	532.5
Helicopter	16,4,4	5	18, 3	6,40	136	sat	1.2
		5	24, 4	6,40	166	unsat	0.61
		7	24, 4	9, 56	213	sat	8.2
		9	36, 6	6, 72	402	sat	24.5
		12	24,4	6, 96,	338	sat	60.6
		15	24, 4	6, 96,	576	sat	158.8
		18	24, 4	10, 96,	640	–	–

Table 2: Experimental results for *Synthesis*

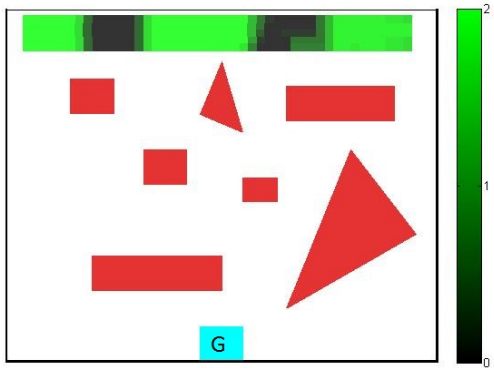

Figure 3: Vulnerability Analysis of Initial States. Adversary may cause the system to hit an obstacle or delay the time of reaching beyond T

Notice that (14) is essentially the same as (2) by switching the roles of $\mathbf{u}$ and $\mathbf{a}$, and negating *Safe* to get *Unsafe*.

We suppose that the set of adversarial input *Adv* is a polytopic set and the control $Ctr = \{\mathbf{u} \in \mathcal{U}^T : \sum_{t\in T} ||u_t||^2 \leq b\}$ is specified by budget $b \geq 0$. For general convex compact sets *Ctr* and *Adv*, one can come up with an under approximated *Adv* as polytopic set and an over-approximated *Ctr* with budget b. As we discuss in Section 5.1, this approximation is sound.

We synthesize a look-up table $\{(\mathcal{I}_i, \mathbf{a}_i)\}_i$ as the strategy of the adversary, such that (i) $\mathcal{I}_i \subseteq \mathcal{X}$, and (ii) for each state $x \in \mathcal{I}_i$, the corresponding adversary $\mathbf{a}_i$ satisfies (14). During the evolution of the plant under controller, the adversary act only when the system reaches a state $x \in \mathcal{I}_i$ for some $\mathcal{I}_i$ in the look-up table, then the corresponding attack $\mathbf{a}_i$ is triggered at x which breaks the safety of the system.

The synthesis of attacks uses similar idea of creating covers of the states as in *TableSynthesis* without refinements.

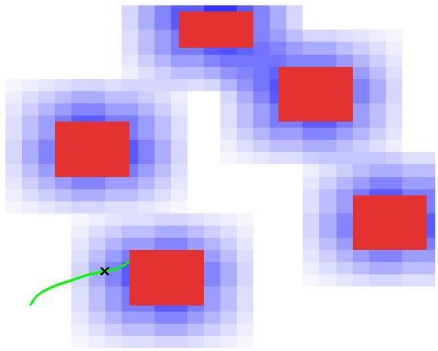

Figure 4: Attack Generation. The darker a region is, a larger portion of velocity is vulnerable. If the vehicle visit a region near to an obstacle, it could survive only if its initial velocity is pointing outwards.

Suppose the set of states $\mathcal{X} \subseteq \mathbb{R}^4$ is compact. An adversary first creates a uniform cover of the state space, then search for an attack for each cover. If the synthesis succeed and returns an attack $\mathbf{a}$, then the cover is *vulnerable* and is stored in the look-up table of attacks paired with the attack $\mathbf{a}$.

A result of the synthesis is illustrated in 4, where the red boxes specify obstacles. The vulnerable covers, each of which is a subset of $\mathbb{R}^4$, are projected on the 2-D plane and visualized as blue regions, where the white region are not vulnerable to attackers. The darkness of a region corresponds to the number of vulnerable covers have projection in the region. That is, if the vehicle is in a dark region, a large portion of its velocity space is vulnerable under attacks that makes the system unsafe. A sample trajectory is captured by the green curve, where, as it enters light shadow region, its velocity does not fall into a vulnerable cover right away. As it approach further, it enters a vulnerable cover and an attack is triggered at the point with cross mark.

7. CONCLUSION

We present a controller synthesis algorithm for a discrete time reach-avoid problem in the presence of adversaries. Specifically, we present a sound and complete algorithm for the case with linear time-varying dynamics and an adversary with a budget on the total L2-norm of its actions. The algorithm combines techniques in control theory and synthesis approaches coming from formal method and programming language researches. Our approach first precisely converts the reach set of the complete system into a composition of non-determinism from the adversary input and the choice of initial state, and an adversary-free trajectory with fixed initial state. Then we enhance the *Safe* and *Goal* conditions by solving a sequence of quadratic-constrained linear optimization problem. And finally we derive a linear quantifier-free SMT formula for the adversary-free trajectories, which can be solved effectively by SMT solvers. The algorithm is then extended to solve problems with more general initial set and constraints of controller and adversary. We present preliminary experimental results that show the effectiveness of this approach on several example problems. The algorithm synthesizes adversary-resilient controls for a 4-dimensional system for 320 rounds and for a 16-dimensional system for 15 rounds in minutes. The algorithm is extended to analyze vulnerability of states and to synthesize attacks.

Future Direction

There are several interesting follow-up research topics. For example, the solution of linear $ARAC$ can be used to solve adversary-free nonlinear avoid-reach problems, where the dynamics can be linearized along a nominal trajectory and the linearization error is modeled as adversary.

We also planned to extend the approach to synthesize switched controller for infinite horizon by applying a similar approach as suggested in [25].

Another interesting direction is to precisely define a dual problem of the linear $ARAC$. Since reachability is dual to detectability, we envision that there exists a detectability type problem dual to $ARAC$, such that the adversary adds noise to the measurements. The question is then how well we can estimate whether the system is in unsafe state based on the noisy measurements.

8. REFERENCES

[1] S. P. Bhattacharyya, H. Chapellat, and L. H. Keel, "Robust control," *The Parametric Approach, by Prentice Hall PTR*, 1995.

[2] T. Basar, G. J. Olsder, G. Clsder, T. Basar, T. Baser, and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1995, vol. 200.

[3] P. Tabuada and G. J. Pappas, "Model checking ltl over controllable linear systems is decidable," in *Hybrid systems: computation and control*. Springer, 2003, pp. 498–513.

[4] A. Ulusoy, T. Wongpiromsarn, and C. Belta, "Incremental controller synthesis in probabilistic environments with temporal logic constraints," *The International Journal of Robotics Research*, p. 0278364913519000, 2014.

[5] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 5319–5325. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2014.6907641

[6] Z. Zhou, R. Takei, H. Huang, and C. J. Tomlin, "A general, open-loop formulation for reach-avoid games." in *CDC*, 2012, pp. 6501–6506.

[7] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems." in *HotSec*, 2008.

[8] F. Pasqualetti, F. Dorfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 11, pp. 2715–2729, Nov 2013.

[9] J. Dahl and L. Vandenberghe, "Cvxopt: A python package for convex optimization," in *Proc. eur. conf. op. res*, 2006.

[10] L. De Moura and N. Bjørner, *Z3: An efficient SMT solver*. Springer, 2008.

[11] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, 2009.

[12] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," in *Hybrid Systems: Computation and Control*. Springer, 2009, pp. 31–45.

[13] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*. ACM, 2011, pp. 355–366.

[14] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure estimation and control for cyber-physical systems under adversarial attacks," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1454–1467, June 2014.

[15] Y. Shoukry, J. Araujo, P. Tabuada, M. Srivastava, and K. H. Johansson, "Minimax control for cyber-physical systems under network packet scheduling attacks," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*. ACM, 2013, pp. 93–100.

[16] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, "Smt-based synthesis of integrated task and motion plans from plan outlines."

[17] T. Beyene, S. Chaudhuri, C. Popeea, and A. Rybalchenko, "A constraint-based approach to solving games on infinite graphs," in *Proceedings of the 41st annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 2014, pp. 221–234.

[18] J. Ding, E. Li, H. Huang, and C. J. Tomlin, "Reachability-based synthesis of feedback policies for motion planning under bounded disturbances," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2160–2165.

[19] B. Mettler, T. Kanade, and M. B. Tischler, *System identification modeling of a model-scale helicopter*. Carnegie Mellon University, The Robotics Institute, 2000.

[20] O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *Hybrid Systems: Computation and Control*. Springer, 2000, pp. 73–88.

[21] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, "Beyond hytech: Hybrid systems analysis using interval numerical methods," pp. 130–144, 2000.

[22] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

[23] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[24] B. Dutertre and L. De Moura, "A fast linear-arithmetic solver for dpll (t)," in *Computer Aided Verification*. Springer, 2006, pp. 81–94.

[25] J.-W. Lee, "Inequality-based properties of detectability and stabilizability of linear time-varying systems in discrete time," *Automatic Control, IEEE Transactions on*, vol. 54, no. 3, pp. 634–641, March 2009.